CHAIR OF DECENTRALIZED INFORMATION SYSTEMS & DATA MANAGEMENT

TECHNISCHE UNIVERSITY MUNICH

Thesis proposal

Scalabe range lock

Author:	**Thua Duc Nguyen**
Supervisor:	**Prof. Dr. Viktor Leis**
Advisor:	**Ph.D Lam Duy Nguyen**

1 Introduction

Scalable range locks is a synchronization mechanism designed to enable parallelism for non-overlapping ranges. Originally designed for file systems, they have received considerable attention[1] in the Linux kernel community for their potential to solving the bottlenecks in the virtual memory management subsystem. These locks provide a means for multiple threads or processes to simultaneously access disjoint segments of a shared resource, thereby improving system performance and concurrency.

2 Motivation

In recent years, there has been a surge of interest in range locks in a different context. A good example is that the Linux kernel community is considering using range locking to replace bottleneck of mmap_lock[2]. The mmap_lock is a reader/writer lock that controls access to a process's address space. Before making any changes there, the kernel must acquire that lock. Pagefault handling must also acquire mmap_lock to ensure that the address space doesn't change in surprising ways while a fault is being resolved. A process can have a large address space and many threads running concurrently, turning mmap_lock into a significant bottleneck.

3 Related Work

Previous research has explored various approaches to concurrency control in distributed systems. Traditional locking mechanisms, such as fine-grained and coarse-grained locks, have been widely studied but often face scalability limitations. More recent approaches, such as distributed locking protocols and range-based access management techniques, offer promising solutions but may lack efficiency and scalability in certain scenarios.

4 Approach

The proposed approach involves designing and implementing a scalable range locking mechanism tailored for distributed systems. The key components of the approach include:

- Designing an algorithm that efficiently manages concurrent accesses to data ranges while minimizing contention and synchronization overheads.
- Leveraging distributed data structures and decentralized coordination techniques to ensure scalability and fault tolerance.
- Developing a flexible and easy-to-use API for integrating the range locking mechanism into existing distributed systems seamlessly.

5 Evaluation

The proposed approach will be evaluated through extensive experimentation and benchmarking under various workload scenarios. The evaluation criteria include:

- Performance: Measure the scalability and throughput of the range locking mechanism under increasing load and concurrent accesses.
- Correctness: Validate the consistency and correctness of data accesses, especially in scenarios involving overlapping data ranges and concurrent operations.
- Scalability: Assess the scalability limits of the range locking mechanism in terms of the number of clients, data range sizes, and system resources.
- Comparison: Compare the performance and scalability of the proposed mechanism against existing locking schemes and state-of-the-art approaches.

6 Expected Outcome

The expected outcome of this research is a scalable range locking mechanism that significantly improves the efficiency and scalability of concurrency control in distributed systems. The findings from the evaluation will provide insights into the performance characteristics, scalability limits, and potential trade-offs of the proposed mechanism, contributing to the advancement of distributed computing research and practice.

7 Resources

For the evaluation of the proposed mechanism, the following resources are required:

- Budget of 20 cores and 200GB of RAM over two months for experimentation and benchmarking.
- Access to a server with NVIDIA GPU for one week to evaluate performance optimizations.
- Access to a 112 core machine for two weeks to assess scalability limits.

These resources are considered reasonable for conducting thorough evaluations and ensuring the feasibility and effectiveness of the proposed approach.

Bibliography

[1] J. Corbet. “The ongoing search for mmap_lock scalability”. In: *LWN.net* (2022). Accessed: 2024-04-21. URL: `https://lwn.net/Articles/893906/`.

[2] J. Corbet. “Concurrent page-fault handling with per-VMA locks”. In: *LWN.net* (2022). Accessed: 2024-04-21. URL: `https://lwn.net/Articles/906852/`.